**Задание:**

1. Создать файл формата .txt, содержащий несколько строк текста. С помощью пакета java.nio нужно прочитать содержимое файла и вывести данные в стандартный поток вывода.
2. Реализовать копирование файла размером 100 Мб 4 методами:

1) FileInputStream/FileOutputStream

2) FileChannel

3) Apache Commons IO

4) Files class

Замерить затраты по времени и памяти и провести сравнительный анализ.

1. Реализовать функцию нахождения 16-битной контрольной суммы файла с использованием бинарных операций и ByteBuffer.
2. При помощи WatchService реализовать наблюдение за каталогом:

1) При создании нового файла в этом каталоге вывести его название;

2) При изменении файла вывести список изменений(добавленных и удалённых строк);

3) При удалении файла вывести его размер и контрольную сумму (использовать реализацию из задания 3).

Если реализовать пункт 3 не предоставляется возможным – докажите это.

**Описание выполнения работы:**

1. Реализована функция вывода данных из текстового файла с помощью пакета java.nio, Листинг 1. Вывод идёт в стандартный поток вывода, Рисунок 1.

Листинг 1 – Вывод данных файла

|  |  |
| --- | --- |
| |  | | --- | | private static void task1(String s) throws IOException {  Path path = Paths.get(s);  List<String> list = Files.readAllLines(path);  for (String str : list)  System.out.println(str);  }  public class Main {  public static void main(String[] args) throws IOException, InterruptedException {  task1("C:\\Users\\Harlequin\\IdeaProjects\\PR2\\untitled\\test.txt");  File fromFile = new File("C:\\Users\\Harlequin\\IdeaProjects\\PR2\\untitled\\UnityHubSetup.exe");  File toFile = new File("C:\\Users\\Harlequin\\IdeaProjects\\PR2\\untitled\\toFile.exe");  Path fileToDeletePath = Paths.get("C:\\Users\\Harlequin\\IdeaProjects\\PR2\\untitled\\toFile.exe");  } | |

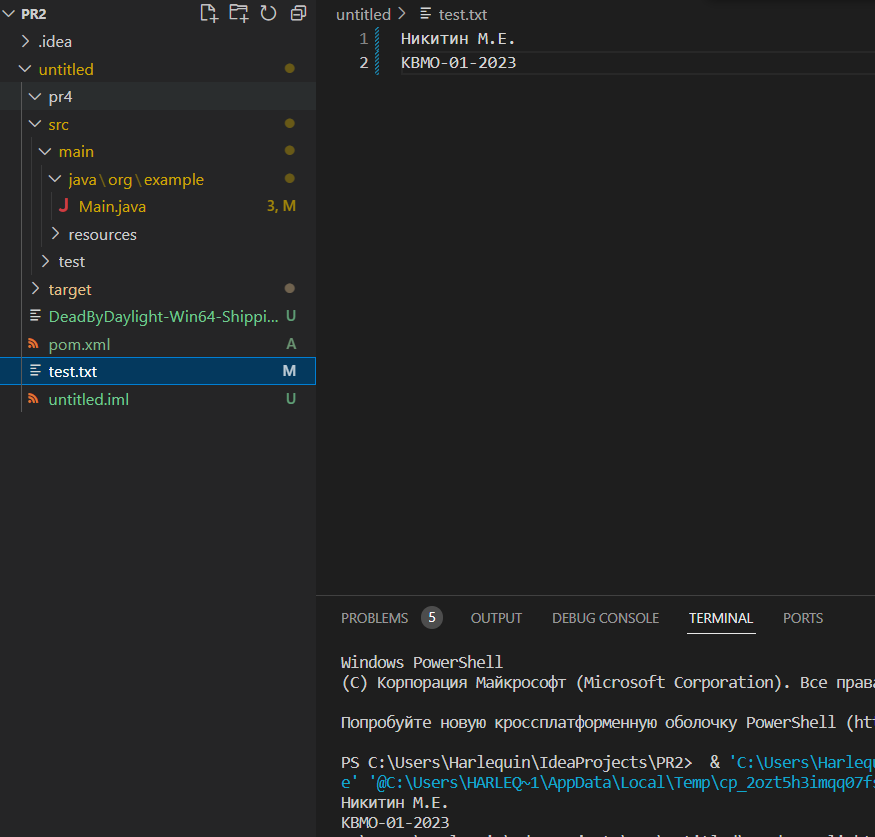


Рисунок 1 – Текстовый документ

1. Для реализации копирования файла были сделаны 4 метода (Листинги 2-5, Рисунок 2):

Листинг 2 – FileInputStream/FileOutputStream

|  |  |
| --- | --- |
| |  | | --- | | private static void method1(File fromFile, File toFile) throws IOException {  InputStream in = new FileInputStream(fromFile);  OutputStream out = new FileOutputStream(toFile);  byte[] buffer = new byte[128000000];  int length;  while ((length = in.read(buffer)) > 0) {  out.write(buffer,0, length);  }  in.close();  out.close();  } | |
| Продолжение листинга 2   |  |  | | --- | --- | | |  | | --- | | startMemory = runtime.totalMemory();  start = System.currentTimeMillis();  method2(fromFile,toFile);  finish = System.currentTimeMillis();  finishMemory = runtime.freeMemory();  System.out.println("Через FileChannel:");  System.out.println("Затраты памяти: " + ((startMemory - finishMemory) / 1024) + " kb");  System.out.println("Затраты времени: " + (finish - start) + " ms\n");  Files.delete(fileToDeletePath); | | |
|  |
| |  | | --- | | private static void method2(File fromFile, File toFile) throws IOException {  FileChannel fc1 = new FileInputStream(fromFile).getChannel();  FileChannel fc2 = new FileOutputStream(toFile).getChannel();  fc2.transferFrom(fc1, 0, fc1.size());  fc1.close();  fc2.close();  }  public class Main {  public static void main(String[] args) throws IOException, InterruptedException {  startMemory = runtime.totalMemory();  start = System.currentTimeMillis();  method2(fromFile,toFile);  finish = System.currentTimeMillis();  finishMemory = runtime.freeMemory();  System.out.println("Через FileChannel:");  System.out.println("Затраты памяти: " + ((startMemory - finishMemory) / 1024) + " kb");  System.out.println("Затраты времени: " + (finish - start) + " ms\n");  Files.delete(fileToDeletePath);  } |   Листинг 3 – FileChannel |
|  |
| |  | | --- | | private static void method3(File fromFile, File toFile) throws IOException {  FileUtils.copyFile(fromFile, toFile);  }  public class Main {  public static void main(String[] args) throws IOException, InterruptedException {  startMemory = runtime.totalMemory();  start = System.currentTimeMillis();  method3(fromFile,toFile);  finish = System.currentTimeMillis();  finishMemory = runtime.freeMemory();  System.out.println("С помощью ApacheCommonsIO:");  System.out.println("Затраты памяти: " + ((startMemory - finishMemory) / 1024) + " kb");  System.out.println("Затраты времени: " + (finish - start) + " ms\n");  Files.delete(fileToDeletePath);} |   Листинг 4 – Apache Commons IO |

|  |
| --- |
| private static void method4(File fromFile, File toFile) throws IOException {  Files.copy(fromFile.toPath(), toFile.toPath());  }  public class Main {  public static void main(String[] args) throws IOException, InterruptedException {  startMemory = runtime.totalMemory();  start = System.currentTimeMillis();  method4(fromFile,toFile);  finish = System.currentTimeMillis();  finishMemory = runtime.freeMemory();  System.out.println("Через класс Files:");  System.out.println("Затраты памяти: " + ((startMemory - finishMemory) / 1024) + " kb");  System.out.println("Затраты времени: " + (finish - start) + " ms\n");  Files.delete(fileToDeletePath);  } |

Листинг 5 – Files class

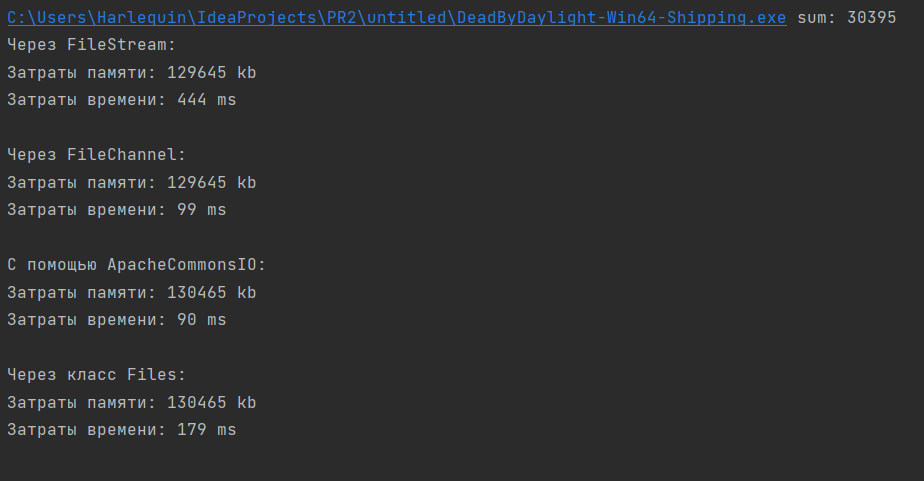


Рисунок 2 – Результат выполнения программы

1. Реализована функция нахождения контрольной суммы с использованием бинарных операций и ByteBuffer. (Листинг 6, Рисунок 3).

Листинг 6 – Функция нахождения суммы sum

|  |  |
| --- | --- |
| |  | | --- | | private static int sum(ByteBuffer bb) {  int sum = 0;  while (bb.hasRemaining()) {  if ((sum & 1) != 0)  sum = (sum >> 1) + 0x8000;  else  sum >>= 1;  sum += bb.get() & 0xff;  sum &= 0xffff;  }  return sum;  }  private static void sum(File file) throws IOException {  // Получение канала из потока  try (FileInputStream fis = new FileInputStream(file);  FileChannel fc = fis.getChannel()) { | |
| Продолжение листинга 6   |  |  | | --- | --- | | |  | | --- | | int size = (int) fc.size();  MappedByteBuffer bb = fc.map(FileChannel.MapMode.READ\_ONLY, 0, size);  System.out.println(file);  System.out.println("Контрольная сумма: " +sum(bb));  }  } | | |

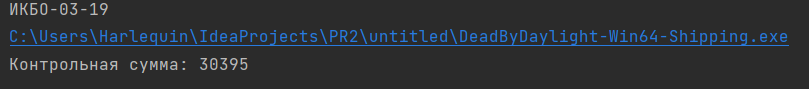


Рисунок 3 – Результат выполнения программы

1. Было реализовано наблюдение за каталогом (Листинг 7, рисунок 4).

Листинг 7 – Наблюдение за каталогом

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | private static void task4() throws IOException, InterruptedException {  WatchService watchService = FileSystems.getDefault().newWatchService();  Path path = Paths.get("C:\\Users\\Harlequin\\IdeaProjects\\PR2\\untitled\\pr4");  path.register(watchService,  StandardWatchEventKinds.ENTRY\_CREATE,  StandardWatchEventKinds.ENTRY\_DELETE,  StandardWatchEventKinds.ENTRY\_MODIFY);  WatchKey key;  while ((key = watchService.take()) != null) {  for (WatchEvent event : key.pollEvents()) {  if (event.kind().toString() == "ENTRY\_CREATE") {  System.out.println("======== ENTRY\_CREATE ========");  System.out.println(event.context());  }  else if (event.kind().toString() == "ENTRY\_MODIFY" && !event.context().toString().contains("git")) { |   Продолжение листинга 7   |  |  | | --- | --- | | |  | | --- | | System.out.println("======== ENTRY\_MODIFY ========" + event.context());  ProcessBuilder builder = new ProcessBuilder("cmd.exe", "/c", "git diff");  builder.redirectOutput(ProcessBuilder.Redirect.INHERIT);  builder.directory(new File("C:\\Users\\Harlequin\\IdeaProjects\\PR2\\untitled\\pr4"));  builder.start();  builder = new ProcessBuilder("cmd.exe", "/c", "git add .");  builder.directory(new File("C:\\Users\\Harlequin\\IdeaProjects\\PR2\\untitled\\pr4"));  builder.start();  }  else if (event.kind().toString() == "ENTRY\_DELETE" && !event.context().toString().contains("git")){  System.out.println("======== ENTRY\_DELETE ========");  //sum(new File("C:\\Users\\Harlequin\\IdeaProjects\\PR2\\untitled\\\pr4" + event.context()));  }  }  key.reset();  }  } | | |
| Рисунок 4 – Результат выполнения программы |

**Вывод:** В данной практической работе получены навыки работы с Java IO и NIO, в частности узнали как с помощью пакета java.nio вывести содержимое файла, изучили методы работы с файлами, такими как FileInputStream, FileChannel, Apache Commons IO, Files class, провели сравнительных анализ этих методов по времени и памяти, изучили работу ByteBuffer при нахождении контрольной суммы и реализовали наблюдение за каталогом с помощью WatchService.